Project Description:

Our project is titled as "Traffic Jam Control System based on User Inputs". In this project we drew a traffic lamp post, a vehicle , a stick figure of person and a zebra crossing. Firstly , when we give '1' as input, the traffic lamp post generates a red light; which means the vehicle stops just before the zebra crossing and the person crosses the road. Secondly when we give '2' as input, the traffic lamp post generates a yellow light; which means the vehicle moves a bit forward but just before the zebra crossing and the person gets a signal of vehicle movement. Finally, when we give '3' as input, the traffic lamp post generates a green light; which means the vehicle moves past the zebra crossing and the person remains standby.

We used the midpoint line algorithm to draw the rectangular body of the vehicle, stick figure of the person and the lamp post. We used the midpoint circle algorithm to draw the wheels of the vehicle, the circular lights of the traffic lamp post and the head part of the stick figure (person). We also used the midpoint line algorithm and the midpoint circle algorithm to fill the color of the lamp post and the circular lights respectively.

We used two transformations- translation and shearing. To enable the person (stick figure) to cross the road, we used translation along the y-axis. Again, to enable the vehicle to move before and past the zebra crossing, we used translation along the x-axis. For the zebra crossing, we used shearing along the x-axis (SHx).

Project Source Code:

```
!pip install -q lucid>=0.2.3

import numpy as np

import ctypes.util
from lucid.misc.gl.glcontext import create_opengl_context

# Now it's safe to import OpenGL and EGL functions
import OpenGL.GL as gl
from OpenGL.GLU import *

#create_opengl_context() #creates GL context that is attached to an
# offscreen surface of specified size. Note that rendering to buffers
# of different size and format is still possible with OpenGL Framebuffers.
#
# Users are expected to directly use EGL calls in case more advanced
# context management is required.
```

```
WIDTH, HEIGHT = 800,600
create_opengl_context((WIDTH, HEIGHT))

# OpenGL context is available here.

print(gl.glGetString(gl.GL_VERSION))
print(gl.glGetString(gl.GL_VENDOR))
#print(gl.glGetString(gl.GL_EXTENSIONS))


def findZone(dx,dy):
    if abs(dx)>abs(dy):
        if dx>0 and dy>0:
            return 0
        elif dx<0 and dy>0:
            return 3
        elif dx<0  and dy<0:
            return 4
        elif dx>0 and dy<0:
            return 7
    else:
        if dx > 0 and dy > 0:
            return 1
        elif dx < 0 and dy > 0:
            return 2
        elif dx < 0 and dy < 0:
            return 5
        elif dx > 0 and dy < 0:
            return 6

def convertZone_0_to_n(zone,x1,y1):
    if zone == 0:
        return x1,y1
    elif zone == 1:
        return y1,x1
    elif zone == 2:
        return -y1,x1
    elif zone == 3:
        return -x1,y1
    elif zone == 4:
```

```
        return -x1,-y1
    elif zone == 5:
        return -y1,-x1
    elif zone == 6:
        return y1,-x1
    elif zone == 7:
        return x1,-y1
def convertZone_n_to_0(zone,x1,y1,x2,y2):
    if zone == 0:
        return x1,y1,x2,y2
    elif zone == 1:
        return y1,x1,y2,x2
    elif zone == 2:
        return y1,-x1,y2,-x2
    elif zone == 3:
        return -x1,y1,-x2,y2
    elif zone == 4:
        return -x1,-y1,-x2,-y2
    elif zone == 5:
        return -y1,-x1,-y2,-x2
    elif zone == 6:
        return -y1,x1,-y2,x2
    elif zone == 7:
        return x1,-y1,x2,-y2



def midPointAlgorithm(zone,x1,y1,x2,y2):
    dx = x2 - x1
    dy = y2 - y1

    D = 2*dy - dx
    D_NE = 2*(dy-dx)
    DE = 2*dy
    x = x1
    y = y1


    # iterate through value of X
    while (x < x2):
```

```
        x = x + 1
        if (D < 0):
            D+=DE
        else:
            D+=D_NE
            y = y + 1
        a,b = convertZone_0_to_n(zone,x,y)
        gl.glPointSize(5)
        gl.glBegin(gl.GL_POINTS)
        gl.glVertex2f(a/1920,b/1080)
        gl.glEnd()



def run(x1,y1,x2,y2):
    dx = x2-x1
    dy = y2-y1
    zone = findZone(dx,dy)
    x1,y1,x2,y2 = convertZone_n_to_0(zone,x1,y1,x2,y2)
    midPointAlgorithm(zone,x1,y1,x2,y2)



run(-1400, -1000, -1401, -200)   #left vertical of stand
run(-1200, -1000, -1201, -200)   #right vertical of stand
run(-1400, -1000, -1200, -1001)

run(-1600, -200, -1000, -201)        #nicher horizontal line
run(-1600, 1000, -1001, 1001)  #uporer horizontal line
run(-1600, -200, -1601, 1001)
run(-1000, -201, -1001, 1001)

i=-1400
while i<=-1200:

    run(  i,-1000, i+800  ,-200)

    i+=20
```

```
def drawPoints(x,y):
    gl.glPointSize(2)
    #gl.glColor3f(1,.1,.1)
    gl.glBegin(gl.GL_POINTS)
    gl.glVertex2f(x/1920,y/1080)
    gl.glEnd()



def midpoint_circle(X,Y,R):
    D = 1 - R
    x = 0
    y = R

    while x < y:
        if D < 0:
            D = D + 2*x + 3
            x+=1
        else:
            D = D + 2*x -2*y +5
            x+=1
            y-=1

        drawPoints(X+y,Y+x) #Zone 0
        drawPoints(X+x,Y+y) #Zone 1
        drawPoints(X-x,Y+y) #Zone 2
        drawPoints(X-y, Y+x) #Zone 3
        drawPoints(X-y,Y-x) #Zone 4
        drawPoints(X-x,Y-y) #Zone 5
        drawPoints(X+x,Y-y) #Zone 6
        drawPoints(X+y,Y-x) #Zone 7


X = -1300
Y = 0
R = 150
gl.glColor3f(.1,1,.1) #Green circle
midpoint_circle(X,Y,R)
```

```
gl.glColor3f(1,1,0.1) #Yellow circle
midpoint_circle(-1300,400,150)

gl.glColor3f(1,.1,.1) #Red circle
midpoint_circle(-1300,800,150)



#Person
gl.glColor3f(1,1,1)
midpoint_circle(250,-500,100) #Head
gl.glColor3f(1,1,1)
run(250, -900, 251, -600) #body
run(250,-650, 350,-750) #left hand
run(251,-651,150,-750)  #right hand
run(250,-900, 350,-967) #right leg
run(251,0,150,-967)  #left leg

#Vehicle
run(-900, 0, -901, 251)
run(-150, 1, -151, 250)
run(-900, 0, -150, 1,)
run(-901, 251, -151, 250)
gl.glColor3f(1,1,1)
midpoint_circle(-700,0,80)
midpoint_circle(-400,0,80)


nosh = input()

#for fill up green circle
if int(nosh)==3:
  for i in range(145,0,-5) :
      gl.glColor3f(.1,1,.1)
      midpoint_circle(-1300,0,i)


#for fill up red circle
if int(nosh)==1:
  for i in range(145,0,-5) :
```

```
      gl.glColor3f(1,.1,.1)
      midpoint_circle(-1300,800,i)


#for fill up yellow circle
if int(nosh)==2:
  for i in range(145,0,-5) :
      gl.glColor3f(1,1,.1)
      midpoint_circle(-1300,400,i)




numDraw = {
    1: [[-1300, 750, -1301, 850]],
    2: [[-1350, 400, -1299, 401], [-1351, 350, -1300, 349],
        [-1351, 300, -1300, 299], [-1299, 400, -1300, 350],
        [-1350, 349, -1349, 299]],
    3: [[-1350, -1, -1299, 0], [-1351, -50, -1300, -49],
        [-1351, -101, -1300, -100], [-1299, 0, -1300, -100]],

}


def loadDigits(n):
    for i in numDraw:
        if i == int(n):
            for k in numDraw[i]:
                gl.glColor3f(.1,.1,.1)
                run(k[0], k[1], k[2], k[3])



#nosh = input()
loadDigits(nosh)


def drawPoint(x,y):
  gl.glVertex2f(x/(WIDTH/2),y/(HEIGHT/2))
```

```
#gl.glPointSize(1.4)

import random as rand

import time  #animate
from random import random,sample
from math import pi,sin,cos

def randColor(): return sample([0,random(),1],3)

from IPython.display import clear_output #render library
from IPython.display import display #render library
from PIL import Image #render library

def render(): #animate
  img_buf = gl.glReadPixelsub(0, 0, WIDTH, HEIGHT, gl.GL_RGB,
gl.GL_UNSIGNED_BYTE)
  img = np.frombuffer(img_buf,np.uint8).reshape(HEIGHT, WIDTH, 3)[::-1]
  display(Image.fromarray(img,'RGB'))


#gl.glClear(gl.GL_COLOR_BUFFER_BIT)


if int(nosh)==3:
    a=300

    p=0.11
    t=9
    for i in range(t):
        gl.glClear(gl.GL_COLOR_BUFFER_BIT)

        t=np.array([[1, 0, a],
                    [0, 1, 0],
                    [0, 0, 1]])

        v1 = np.array([[-151],
                       [250],
                       [1]])
```

```
v2 = np.array([[-150],
               [1],
               [1]])


v3 = np.array([[-900],
               [0],
               [1]])

v4 = np.array([[-901],
               [251],
               [1]])

v5 = np.array([[-700],
               [0],
               [1]])


v6 = np.array([[-400],
               [0],
               [1]])

v11 = np.matmul(t,v1)
v22 = np.matmul(t,v2)
v33 = np.matmul(t,v3)
v44 = np.matmul(t,v4)
v55 = np.matmul(t,v5)
v66 = np.matmul(t,v6)


gl.glColor3f(.1,.1,1)

run(v11[0][0],v11[1][0], v22[0][0],v22[1][0])
run(v22[0][0],v22[1][0], v33[0][0],v33[1][0])
run(v44[0][0],v44[1][0], v11[0][0],v11[1][0])
run(v44[0][0],v33[1][0], v33[0][0],v44[1][0])
gl.glColor3f(.1,.1,1)
midpoint_circle(v55[0][0], v55[1][0], 80)
```

```
midpoint_circle(v66[0][0], v66[1][0], 80)


#shearing of zebra crossing
s=np.array([[1, p, 0],
            [0, 1, 0],
            [0, 0, 1]])

z1 = np.array([[551],
               [200],
               [1]])
z2 = np.array([[550],
               [1],
               [1]])
z3 = np.array([[0],
               [0],
               [1]])

z4 = np.array([[1],
               [201],
               [1]])
s11 = np.matmul(s,z1)
s22 = np.matmul(s,z2)
s33 = np.matmul(s,z3)
s44 = np.matmul(s,z4)
gl.glColor3f(1,1,1)
run(s44[0][0],s44[1][0], s11[0][0],s11[1][0])
run(s22[0][0],s22[1][0], s11[0][0],s11[1][0])
run(s33[0][0],s33[1][0], s22[0][0],s22[1][0])
run(s33[0][0],s33[1][0], s44[0][0],s44[1][0])

z5 = np.array([[551],
               [450],
               [1]])
z6 = np.array([[550],
               [251],
               [1]])
z7 = np.array([[0],
               [250],
               [1]])
```

```
z8 = np.array([[1],
               [451],
               [1]])
s55 = np.matmul(s,z5)
s66 = np.matmul(s,z6)
s77 = np.matmul(s,z7)
s88 = np.matmul(s,z8)
gl.glColor3f(1,1,1)
run(s88[0][0],s88[1][0], s55[0][0],s55[1][0])
run(s66[0][0],s66[1][0], s55[0][0],s55[1][0])
run(s77[0][0],s77[1][0], s66[0][0],s66[1][0])
run(s77[0][0],s77[1][0], s88[0][0],s88[1][0])


z9 = np.array([[551],
               [700],
               [1]])
z10 = np.array([[550],
               [501],
               [1]])
z11 = np.array([[0],
               [500],
               [1]])

z12 = np.array([[1],
               [701],
               [1]])
s99 = np.matmul(s,z9)
s1010 = np.matmul(s,z10)
s1111 = np.matmul(s,z11)
s1212 = np.matmul(s,z12)
gl.glColor3f(1,1,1)
run(s1212[0][0],s1212[1][0], s99[0][0],s99[1][0])
run(s1010[0][0],s1010[1][0], s99[0][0],s99[1][0])
run(s1111[0][0],s1111[1][0], s1010[0][0],s1010[1][0])
run(s1111[0][0],s1111[1][0], s1212[0][0],s1212[1][0])

z13 = np.array([[551],
               [-50],
```

```
                [1]])
z14 = np.array([[550],
                [-249],
                [1]])
z15 = np.array([[0],
                [-250],
                [1]])

z16 = np.array([[1],
                [-49],
                [1]])
s1313 = np.matmul(s,z13)
s1414 = np.matmul(s,z14)
s1515 = np.matmul(s,z15)
s1616 = np.matmul(s,z16)
gl.glColor3f(1,1,1)
run(s1616[0][0],s1616[1][0], s1313[0][0],s1313[1][0])
run(s1414[0][0],s1414[1][0], s1313[0][0],s1313[1][0])
run(s1515[0][0],s1515[1][0], s1414[0][0],s1414[1][0])
run(s1515[0][0],s1515[1][0], s1616[0][0],s1616[1][0])




#blabla
##for lightpost
gl.glColor3f(.211,.211,.211)
run(-1400, -1000, -1401, -200)   #left vertical of stand
run(-1200, -1000, -1201, -200)   #right vertical of stand
run(-1400, -1000, -1200, -1001)   #uporer connected line
gl.glColor3f(.211,.211,.211)
run(-1600, -200, -1000, -201)   #nicher horizontal line
run(-1600, 1000, -1001, 1001)  #uporer horizontal line
run(-1600, -200, -1601, 1001)  #  left    vertical line
run(-1000, -201, -1001, 1001)   #right vertical line
```

```
X = -1300
Y = 0
R = 150
gl.glColor3f(.1,1,.1) #Green circle
midpoint_circle(X,Y,R)

gl.glColor3f(1,1,0.1) #Yellow circle
midpoint_circle(-1300,400,150)

gl.glColor3f(1,.1,.1) #Red circle
midpoint_circle(-1300,800,150)
i=-1400
while i<=-1200:
  gl.glColor3f(.211,.211,.211)
  run(  i,-1000, i+1  ,-200)
  i+=20

#Person
gl.glColor3f(1,1,1)
midpoint_circle(250,-500,100) #Head
gl.glColor3f(1,1,1)
run(250, -900, 251, -600) #body
run(250,-650, 350,-750) #left hand
run(251,-651,150,-750)  #right hand
run(250,-867, 350,-967) #right leg
run(251,-868,150,-967)  #left leg




for i in range(145,0,-5) :
    gl.glColor3f(.1,1,.1)
    midpoint_circle(-1300,0,i)



#print 3
loadDigits(nosh)
```

```
        render() #animate
        time.sleep(0.7) #animate
        clear_output(wait=True) #this actually refreshes the window

        #gl.glClear(gl.GL_COLOR_BUFFER_BIT)

        a=a+200



elif (int(nosh)==1):
  b= 50
  t=15
  p=0.13
  for i in range(t):
        gl.glClear(gl.GL_COLOR_BUFFER_BIT)

        t=np.array([[1, 0, 0],
                    [0, 1, b],
                    [0, 0, 1]])

        v1 = np.array([[250],
                       [-500],
                       [1]])

        v2 = np.array([[251],
                       [-600],
                       [1]])
        v3 = np.array([[250],
                       [-900],
                       [1]])
        v4 = np.array([[350],     #left hand
                       [-750],
                       [1]])
        v5 = np.array([[250],      #left hand
                       [-650],
                       [1]])
        v6 = np.array([[150],     #right hand
                       [-750],
                       [1]])
```

```
v7 = np.array([[251],     #right hand
               [-651],
               [1]])
v8 = np.array([[350],      #right leg
               [-900],
               [1]])
v9 = np.array([[150],      #right leg
               [-899],
               [1]])
```

```
v11 = np.matmul(t,v1)
v22 = np.matmul(t,v2)   #body
v33 = np.matmul(t,v3)   #body
v44 = np.matmul(t,v4)   #left hand
v55 = np.matmul(t,v5)   #left hand
v66 = np.matmul(t,v6)   #right hand
v77 = np.matmul(t,v7)   #right hand
v88 = np.matmul(t,v8)   #leg
v99 = np.matmul(t,v9)   #leg
```

```
gl.glColor3f(1,1,1)
run(v33[0][0],v33[1][0], v22[0][0],v22[1][0])   #body
run(v55[0][0],v55[1][0], v44[0][0],v44[1][0])  #left hand
run(v77[0][0],v77[1][0], v66[0][0],v66[1][0])  #right hand
run(v99[0][0],v99[1][0], v88[0][0],v88[1][0])  #leg
midpoint_circle(v11[0][0], v11[1][0], 100)       #head

#shearing of zebra crossing
s=np.array([[1, p, 0],
            [0, 1, 0],
            [0, 0, 1]])
```

```
z1 = np.array([[551],
               [200],
               [1]])
z2 = np.array([[550],
               [1],
               [1]])
z3 = np.array([[0],
               [0],
               [1]])

z4 = np.array([[1],
               [201],
               [1]])
s11 = np.matmul(s,z1)
s22 = np.matmul(s,z2)
s33 = np.matmul(s,z3)
s44 = np.matmul(s,z4)
gl.glColor3f(1,1,1)
run(s44[0][0],s44[1][0], s11[0][0],s11[1][0])
run(s22[0][0],s22[1][0], s11[0][0],s11[1][0])
run(s33[0][0],s33[1][0], s22[0][0],s22[1][0])
run(s33[0][0],s33[1][0], s44[0][0],s44[1][0])

z5 = np.array([[551],
               [450],
               [1]])
z6 = np.array([[550],
               [251],
               [1]])
z7 = np.array([[0],
               [250],
               [1]])

z8 = np.array([[1],
               [451],
               [1]])
s55 = np.matmul(s,z5)
s66 = np.matmul(s,z6)
s77 = np.matmul(s,z7)
```

```
s88 = np.matmul(s,z8)
gl.glColor3f(1,1,1)
run(s88[0][0],s88[1][0], s55[0][0],s55[1][0])
run(s66[0][0],s66[1][0], s55[0][0],s55[1][0])
run(s77[0][0],s77[1][0], s66[0][0],s66[1][0])
run(s77[0][0],s77[1][0], s88[0][0],s88[1][0])


z9 = np.array([[551],
               [700],
               [1]])
z10 = np.array([[550],
               [501],
               [1]])
z11 = np.array([[0],
               [500],
               [1]])

z12 = np.array([[1],
               [701],
               [1]])
s99 = np.matmul(s,z9)
s1010 = np.matmul(s,z10)
s1111 = np.matmul(s,z11)
s1212 = np.matmul(s,z12)
gl.glColor3f(1,1,1)
run(s1212[0][0],s1212[1][0], s99[0][0],s99[1][0])
run(s1010[0][0],s1010[1][0], s99[0][0],s99[1][0])
run(s1111[0][0],s1111[1][0], s1010[0][0],s1010[1][0])
run(s1111[0][0],s1111[1][0], s1212[0][0],s1212[1][0])

z13 = np.array([[551],
               [-50],
               [1]])
z14 = np.array([[550],
               [-249],
               [1]])
z15 = np.array([[0],
               [-250],
               [1]])
```

```
 z16 = np.array([[1],
                 [-49],
                 [1]])
 s1313 = np.matmul(s,z13)
 s1414 = np.matmul(s,z14)
 s1515 = np.matmul(s,z15)
 s1616 = np.matmul(s,z16)
 gl.glColor3f(1,1,1)
 run(s1616[0][0],s1616[1][0], s1313[0][0],s1313[1][0])
 run(s1414[0][0],s1414[1][0], s1313[0][0],s1313[1][0])
 run(s1515[0][0],s1515[1][0], s1414[0][0],s1414[1][0])
 run(s1515[0][0],s1515[1][0], s1616[0][0],s1616[1][0])


#blabla
 ##for lightpost
 gl.glColor3f(.211,.211,.211)
 run(-1400, -1000, -1401, -200)   #left vertical of stand
 run(-1200, -1000, -1201, -200)   #right vertical of stand
 run(-1400, -1000, -1200, -1001)   #uporer connected line
 gl.glColor3f(.211,.211,.211)
 run(-1600, -200, -1000, -201)   #nicher horizontal line
 run(-1600, 1000, -1001, 1001)  #uporer horizontal line
 run(-1600, -200, -1601, 1001)  #  left    vertical line
 run(-1000, -201, -1001, 1001)   #right vertical line


 X = -1300
 Y = 0
 R = 150
 gl.glColor3f(.1,1,.1) #Green circle
 midpoint_circle(X,Y,R)

 gl.glColor3f(1,1,0.1) #Yellow circle
 midpoint_circle(-1300,400,150)

 gl.glColor3f(1,.1,.1) #Red circle
 midpoint_circle(-1300,800,150)
 i=-1400
```

```
        while i<=-1200:
          gl.glColor3f(.211,.211,.211)
          run(  i,-1000, i+1  ,-200)
          i+=20

        #Vehicle
        gl.glColor3f(.1,.1,1)
        run(-900, 0, -901, 251)
        run(-150, 1, -151, 250)
        run(-900, 0, -150, 1,)
        run(-901, 251, -151, 250)
        gl.glColor3f(.1,.1,1)
        midpoint_circle(-700,0,80)
        midpoint_circle(-400,0,80)

        if int(nosh)==1:

          for i in range(145,0,-5) :
              gl.glColor3f(1,.1,.1)
              midpoint_circle(-1300,800,i)



        #print 1
        loadDigits(nosh)
        render() #animate
        time.sleep(0.7) #animate
        clear_output(wait=True) #this actually refreshes the window

        #gl.glClear(gl.GL_COLOR_BUFFER_BIT)

        b=b+170


elif(int(nosh)==2):
  p=0.13
  c=100
  gl.glClear(gl.GL_COLOR_BUFFER_BIT)

  t=np.array([[1, 0, c],
```

```
            [0, 1, 0],
            [0, 0, 1]])

v1 = np.array([[-151],
               [250],
               [1]])


v2 = np.array([[-150],
               [1],
               [1]])


v3 = np.array([[-900],
               [0],
               [1]])

v4 = np.array([[-901],
               [251],
               [1]])

v5 = np.array([[-700],
               [0],
               [1]])


v6 = np.array([[-400],
               [0],
               [1]])

v11 = np.matmul(t,v1)
v22 = np.matmul(t,v2)
v33 = np.matmul(t,v3)
v44 = np.matmul(t,v4)
v55 = np.matmul(t,v5)
v66 = np.matmul(t,v6)


gl.glColor3f(.1,.1,1)
```

```
run(v11[0][0],v11[1][0], v22[0][0],v22[1][0])
run(v22[0][0],v22[1][0], v33[0][0],v33[1][0])
run(v44[0][0],v44[1][0], v11[0][0],v11[1][0])
run(v44[0][0],v33[1][0], v33[0][0],v44[1][0])

gl.glColor3f(.1,.1,1)
midpoint_circle(v55[0][0], v55[1][0], 80)
midpoint_circle(v66[0][0], v66[1][0], 80)
#shearing of zebra crossing
s=np.array([[1, p, 0],
              [0, 1, 0],
              [0, 0, 1]])

z1 = np.array([[551],
                [200],
                [1]])
z2 = np.array([[550],
                [1],
                [1]])
z3 = np.array([[0],
                [0],
                [1]])

z4 = np.array([[1],
                [201],
                [1]])
s11 = np.matmul(s,z1)
s22 = np.matmul(s,z2)
s33 = np.matmul(s,z3)
s44 = np.matmul(s,z4)
gl.glColor3f(1,1,1)
run(s44[0][0],s44[1][0], s11[0][0],s11[1][0])
run(s22[0][0],s22[1][0], s11[0][0],s11[1][0])
run(s33[0][0],s33[1][0], s22[0][0],s22[1][0])
run(s33[0][0],s33[1][0], s44[0][0],s44[1][0])

z5 = np.array([[551],
                [450],
                [1]])
z6 = np.array([[550],
```

```
                [251],
                [1]])
z7 = np.array([[0],
                [250],
                [1]])

z8 = np.array([[1],
                [451],
                [1]])
s55 = np.matmul(s,z5)
s66 = np.matmul(s,z6)
s77 = np.matmul(s,z7)
s88 = np.matmul(s,z8)
gl.glColor3f(1,1,1)
run(s88[0][0],s88[1][0], s55[0][0],s55[1][0])
run(s66[0][0],s66[1][0], s55[0][0],s55[1][0])
run(s77[0][0],s77[1][0], s66[0][0],s66[1][0])
run(s77[0][0],s77[1][0], s88[0][0],s88[1][0])


z9 = np.array([[551],
                [700],
                [1]])
z10 = np.array([[550],
                [501],
                [1]])
z11 = np.array([[0],
                [500],
                [1]])

z12 = np.array([[1],
                [701],
                [1]])
s99 = np.matmul(s,z9)
s1010 = np.matmul(s,z10)
s1111 = np.matmul(s,z11)
s1212 = np.matmul(s,z12)
gl.glColor3f(1,1,1)
run(s1212[0][0],s1212[1][0], s99[0][0],s99[1][0])
run(s1010[0][0],s1010[1][0], s99[0][0],s99[1][0])
```

```
run(s1111[0][0],s1111[1][0], s1010[0][0],s1010[1][0])
run(s1111[0][0],s1111[1][0], s1212[0][0],s1212[1][0])

z13 = np.array([[551],
                [-50],
                [1]])
z14 = np.array([[550],
                [-249],
                [1]])
z15 = np.array([[0],
                [-250],
                [1]])

z16 = np.array([[1],
                [-49],
                [1]])
s1313 = np.matmul(s,z13)
s1414 = np.matmul(s,z14)
s1515 = np.matmul(s,z15)
s1616 = np.matmul(s,z16)
gl.glColor3f(1,1,1)
run(s1616[0][0],s1616[1][0], s1313[0][0],s1313[1][0])
run(s1414[0][0],s1414[1][0], s1313[0][0],s1313[1][0])
run(s1515[0][0],s1515[1][0], s1414[0][0],s1414[1][0])
run(s1515[0][0],s1515[1][0], s1616[0][0],s1616[1][0])

#blabla
##for lightpost
gl.glColor3f(.211,.211,.211)
run(-1400, -1000, -1401, -200)   #left vertical of stand
run(-1200, -1000, -1201, -200)   #right vertical of stand
run(-1400, -1000, -1200, -1001)   #uporer connected line
gl.glColor3f(.211,.211,.211)
run(-1600, -200, -1000, -201)   #nicher horizontal line
run(-1600, 1000, -1001, 1001)  #uporer horizontal line
run(-1600, -200, -1601, 1001)  #  left    vertical line
run(-1000, -201, -1001, 1001)   #right vertical line


X = -1300
```

```
Y = 0
R = 150
gl.glColor3f(.1,1,.1) #Green circle
midpoint_circle(X,Y,R)

gl.glColor3f(1,1,0.1) #Yellow circle
midpoint_circle(-1300,400,150)

gl.glColor3f(1,.1,.1) #Red circle
midpoint_circle(-1300,800,150)
i=-1400
while i<=-1200:
  gl.glColor3f(.211,.211,.211)
  run(  i,-1000, i+1  ,-200)
  i+=20

#Person
gl.glColor3f(1,1,1)
midpoint_circle(250,-500,100) #Head
gl.glColor3f(1,1,1)
run(250, -900, 251, -600) #body
run(250,-650, 350,-750) #left hand
run(251,-651,150,-750)  #right hand
run(250,-867, 350,-967) #right leg
run(251,-868,150,-967)  #left leg




for i in range(145,0,-5) :
  gl.glColor3f(1,1,.1)
  midpoint_circle(-1300,400,i)



#print 2
loadDigits(nosh)
```

```
  render() #animate
  time.sleep(0.7) #animate
  clear_output(wait=True) #this actually refreshes the window
```

```
render()
```